OpenCUDA + MPI

A Framework for Heterogeneous GP-GPU Computing

Kenny Ballou, Boise State University Department of Computer Science
Nilab Mohammad Mousa, Boise State University Department of Computer Science
Mentor: Alark Joshi, PhD. Boise State University Department of Computer Science

Abstract

The introduction and rise of General Purpose Graphics Computing has significantly impacted parallel and high-performance computing. It has introduced challenges when it comes to distributed computing with GPUs. Current solutions target specifics: specific hardware, specific network topology, a specific level of processing. Those restrictions on GPU computing limit scientists and researchers in various ways. The goal of OpenCUDA+MPI project is to develop a framework that allows researchers and scientists to write a general algorithm without the overhead of worrying about the specifics of the hardware and the cluster it will run against while taking full advantage of parallel and distributed computing on GPUs. As work toward the solution continues to progress, we have proven the need for the framework and expect to find the framework enables scientists to focus on their research.

Project Objectives

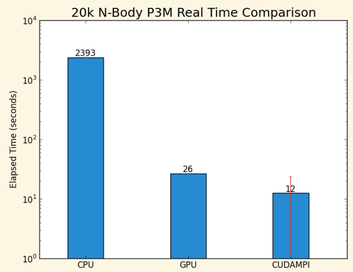
- Develop a framework for highly-parallel computation, utilizing many GPU's over many computers
- Provide cluster administration tools and configurations
- Framework and related code shall be Free and Open-Source software

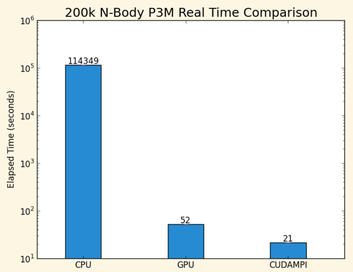
Methodology

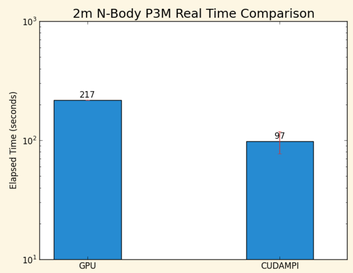
To fully understand the purpose of creating a framework, we first set out to attempt developing parallel code *without* our proposed framework.
We also want to not only compare times of development speed, but actual running time of each solution. As such, we developed a CPU only, a single GPU, and a distributed GPU solution to the N-Body problem and compared running times of each for several different problem sizes.
The N-Body problem was chosen as a test problem because it is *not* an embarassingly parallel problem and, as such, requires inter-communication.

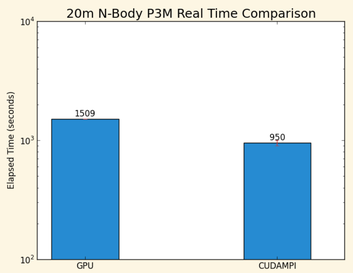
Results

We were able to show that distributing work over a cluster can significanly decrease the amount of time required for a problem. For example, processing the N-Body problem for 200 thousand bodies on a single CPU required 114349 seconds (about 31 hours), as compared to a single GPU which required 52 seconds and distributing over 16 GPU's only took 21 seconds. For more comparisons, please see the table below. [Currently, a comparison between the distributed code with and without the framework is not available.]
Although biased, it was interesting to notice the development time difference between writing the distributed code without the framework as compared to with the framework. Namely, without the framework, it required several weeks, whereas, with the framework, it required less than a week.









Conclusions and Discussion

We can clearly see the increase in performance when we distribute our work over many machines. If our problem is particularly I/O bound, however, we may be causing more slow downs or bottlenecks than worth the computational performance increase.
We need to test the framework more, and with more, different problems. Doing so would highlight areas that need adjustment or re-working. Further, it would show the framework's viability to others and in other problem spaces.

Future Work

- Add better debugging, logging and profiling to the framework
- Expose control of CUDA device initilaization
- Add automatic/ configurable checkpointing
- (Finish) Node Configuration
  - Master/ Head Node Configuration
  - Add real (job) scheduler

Acknowledgments

Special thanks to the Student Research Initative program, Liljana Babinkostova, PhD., and my mentor, Alark Joshi, PhD.